



Scaling and Managing LAMP at Wikimedia



Brion Vibber
CTO, Wikimedia Foundation

Greater San Francisco LAMP Meetup
Santa Clara, CA 2008-10-16

Basic LAMP stack

y'all know the drill...

- Linux
- Apache
- MySQL
- PHP / Perl / Python / Pwhatever

at the core...

keep it simple

Apache+PHP



MySQL

Ahhh, simple is nice.
:)

Simple is slow.

:(

First, add cache!

Squid



Apache+PHP+APC



memcached



MySQL

Squid



Apache+PHP+APC



memcached



MySQL

Good for static-
dynamic sites like a
wiki...



The public face
of a given page
doesn't change
very often, so
you can cache at
the HTTP level.

Seoul

Squid

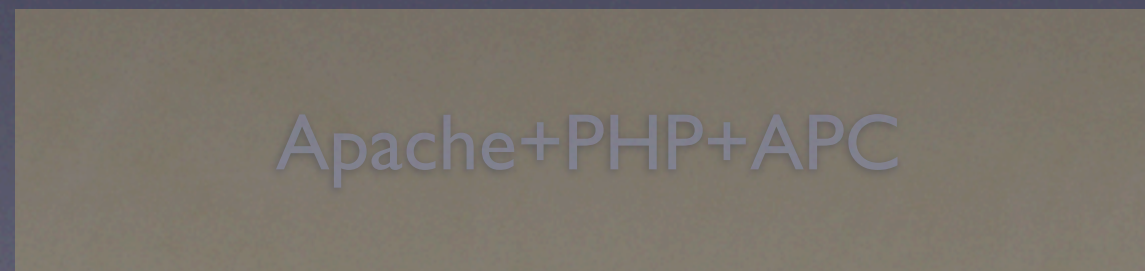
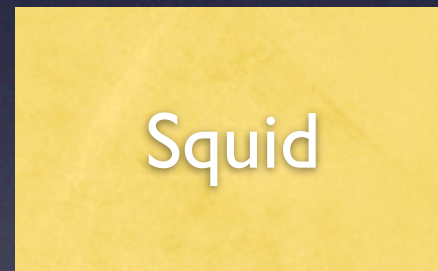
Amsterdam

Squid

Tampa

Squid

Apache+PHP+APC



Good for geographic load balancing, too!

Use cheaper, faster local
bandwidth...

Seoul

Squid

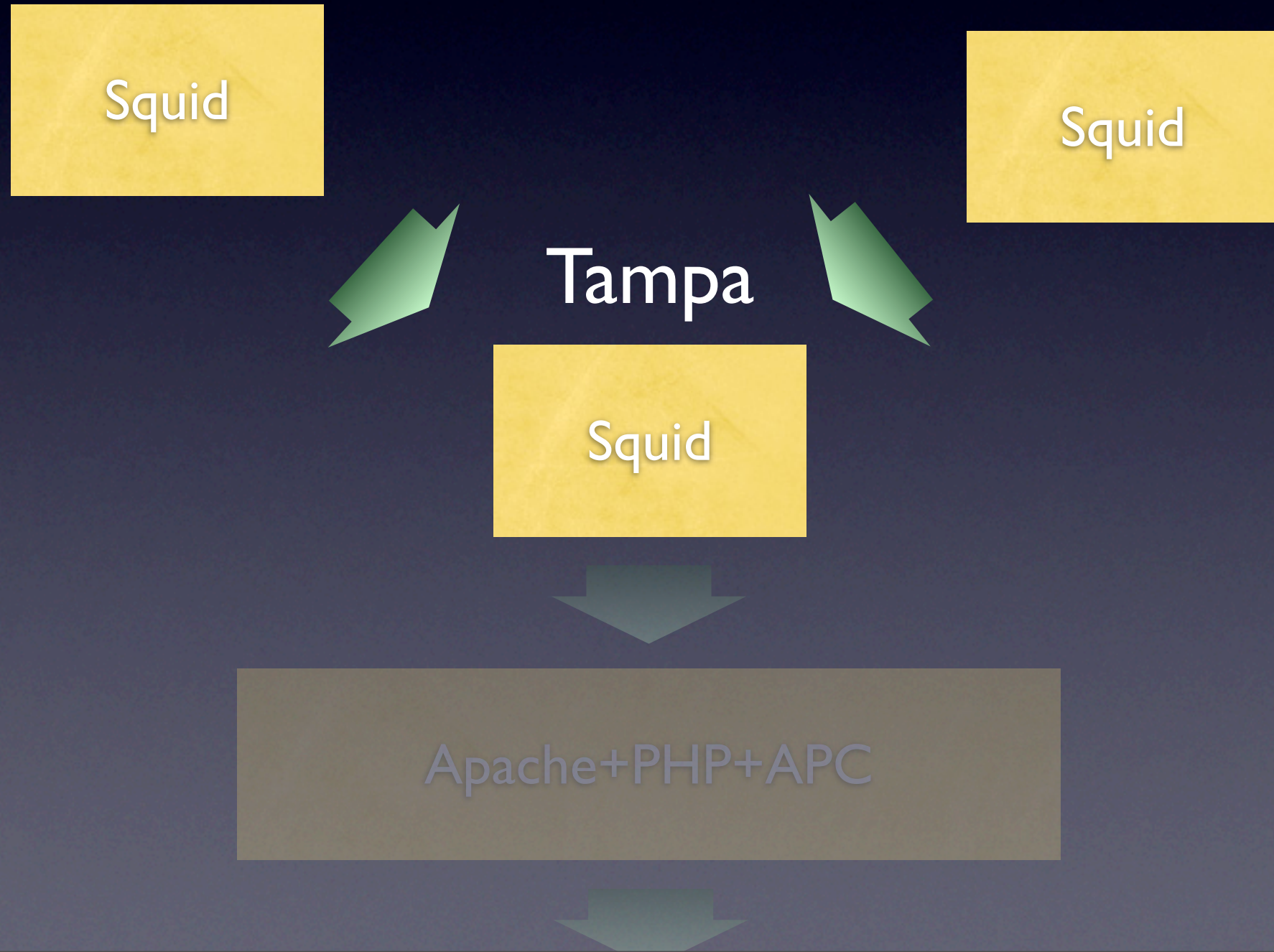
Amsterdam

Squid

Tampa

Squid

Apache+PHP+APC



Squid



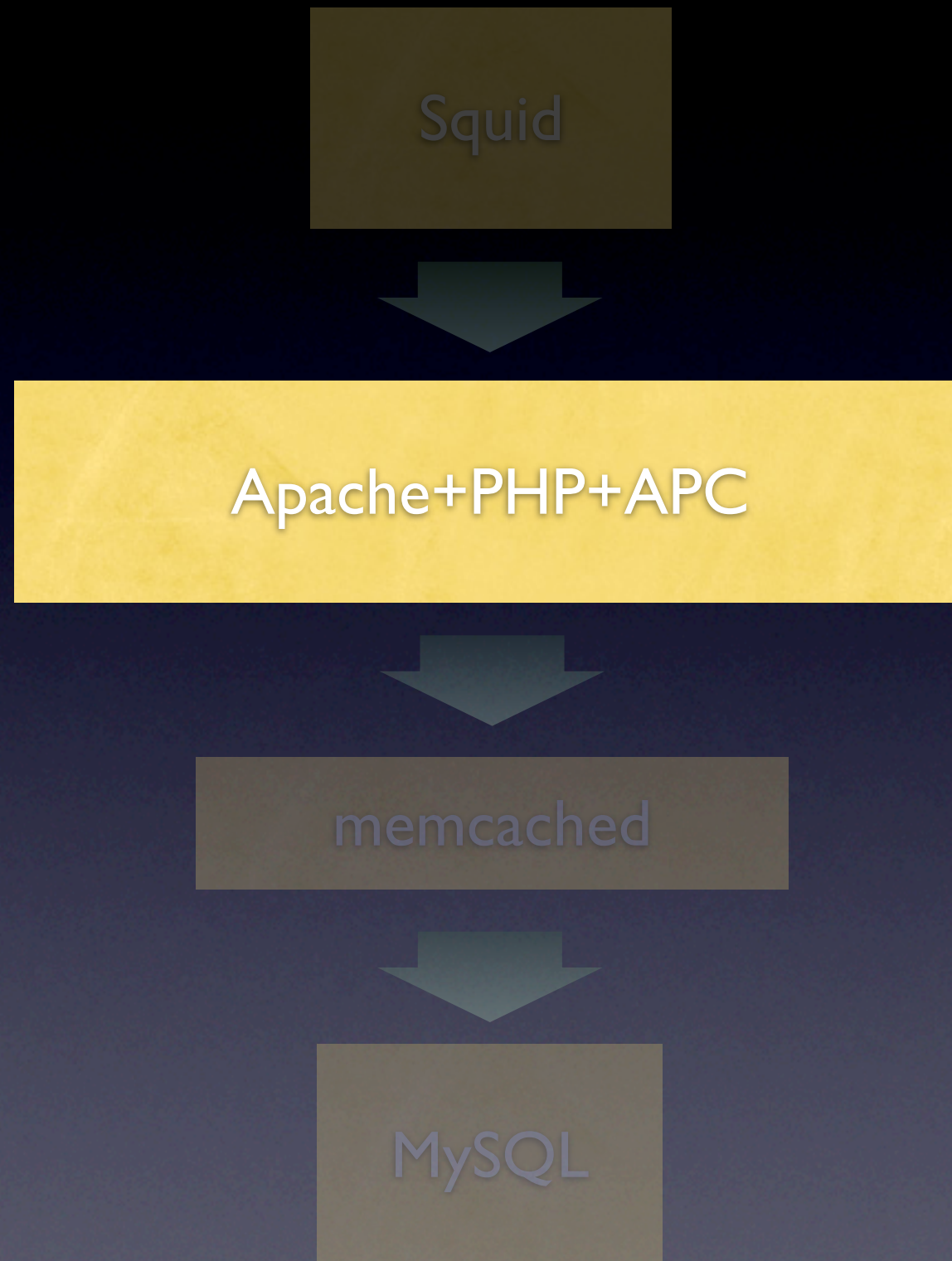
Apache+PHP+APC



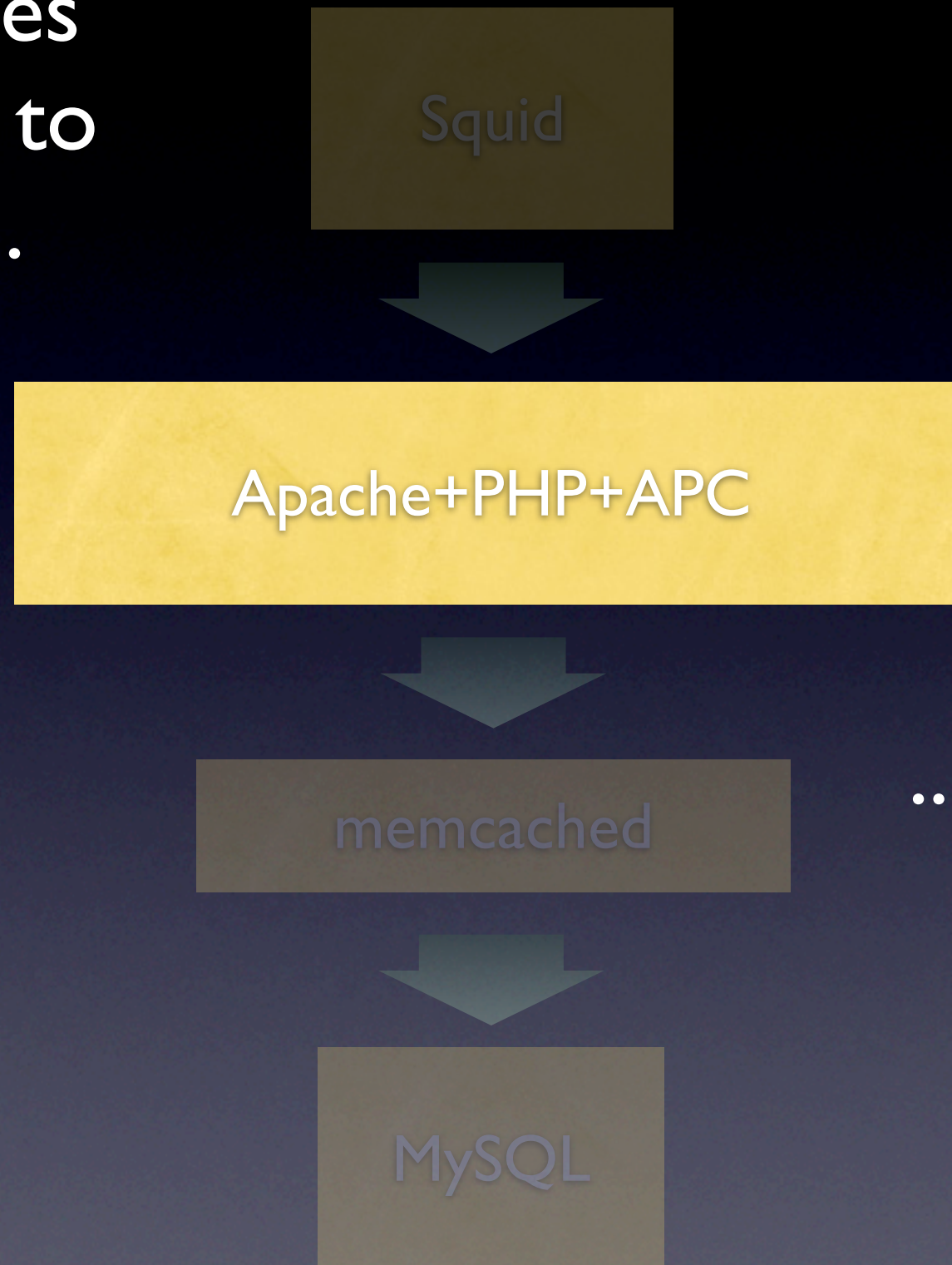
memcached



MySQL



PHP compiles
your scripts to
bytecode...



...then throws it
away after
execution.

Compiling on
every run adds a
lot of
overhead...

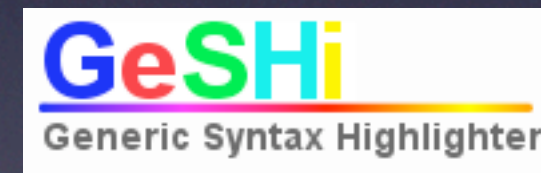
Squid

Apache+PHP+APC

memcached

MySQL

...especially as
your application
grows...



...pulling in lots of
framework and library
code.

Always use an opcode
cache with PHP!

Squid



Apache+PHP+APC



memcached



MySQL

APC

eAccelerator

Zend Platform

...

Drastically reduces
startup time for
large apps, and
moderate
improvements even
for small scripts.

Squid



Apache+PHP+APC



memcached



MySQL

Squid



Apache+PHP+APC



memcached

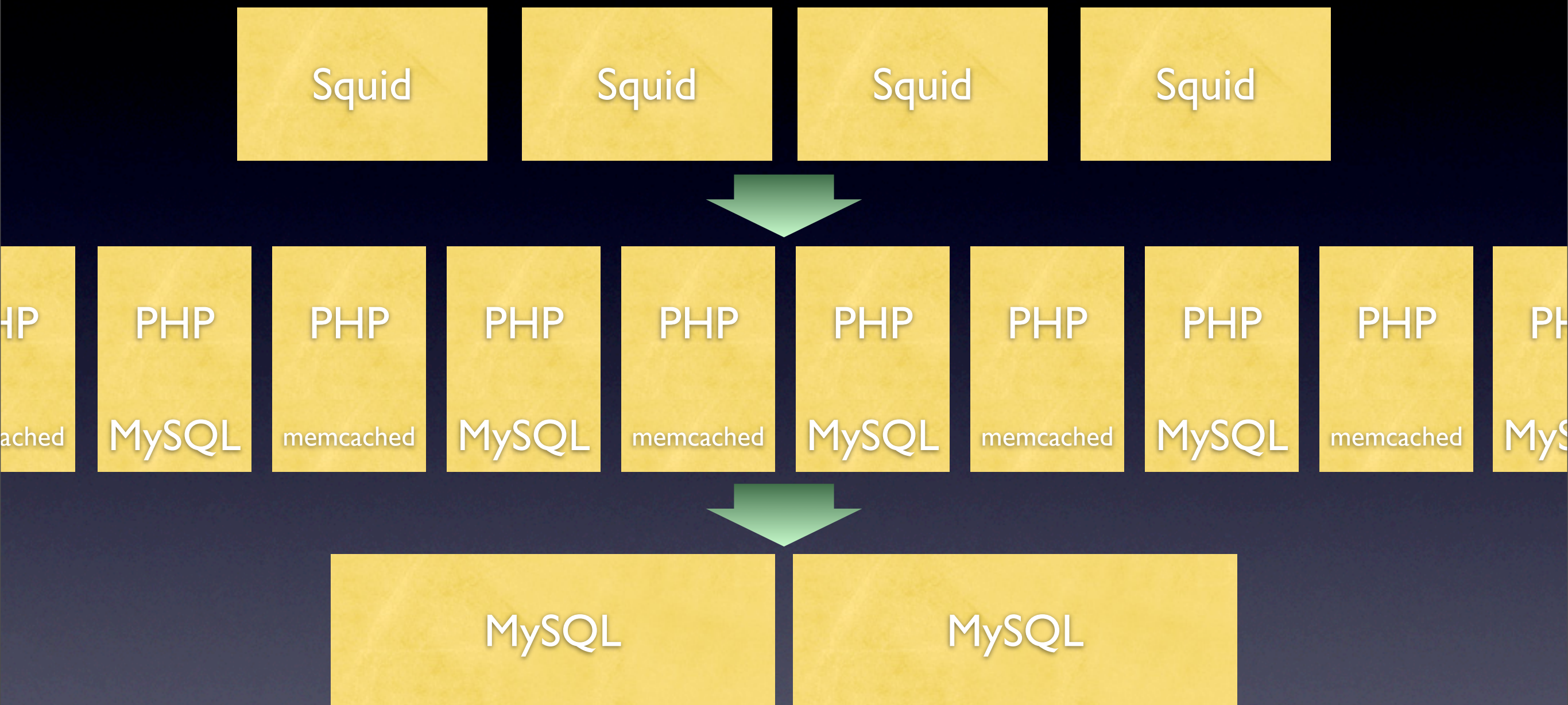


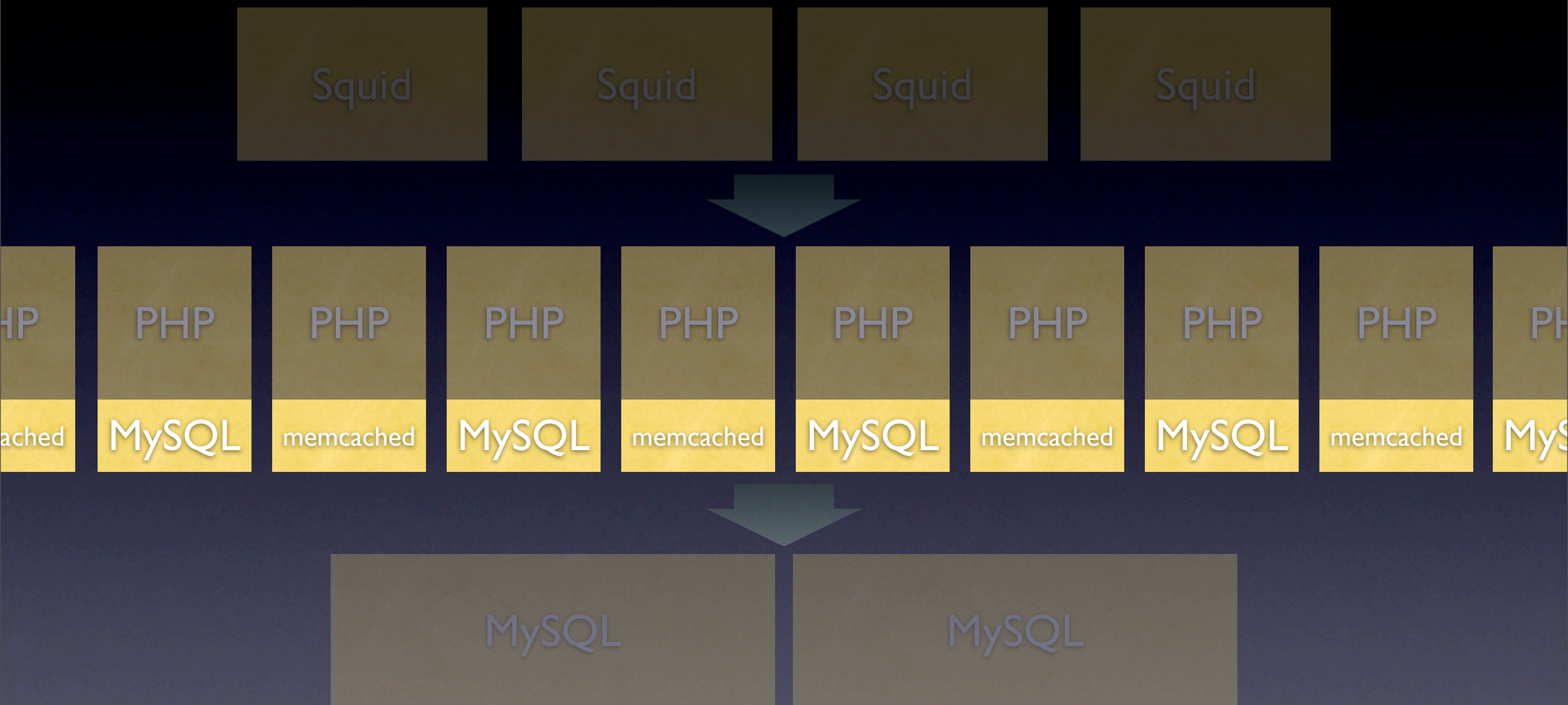
MySQL

Share temporary
data in your
network's
memory

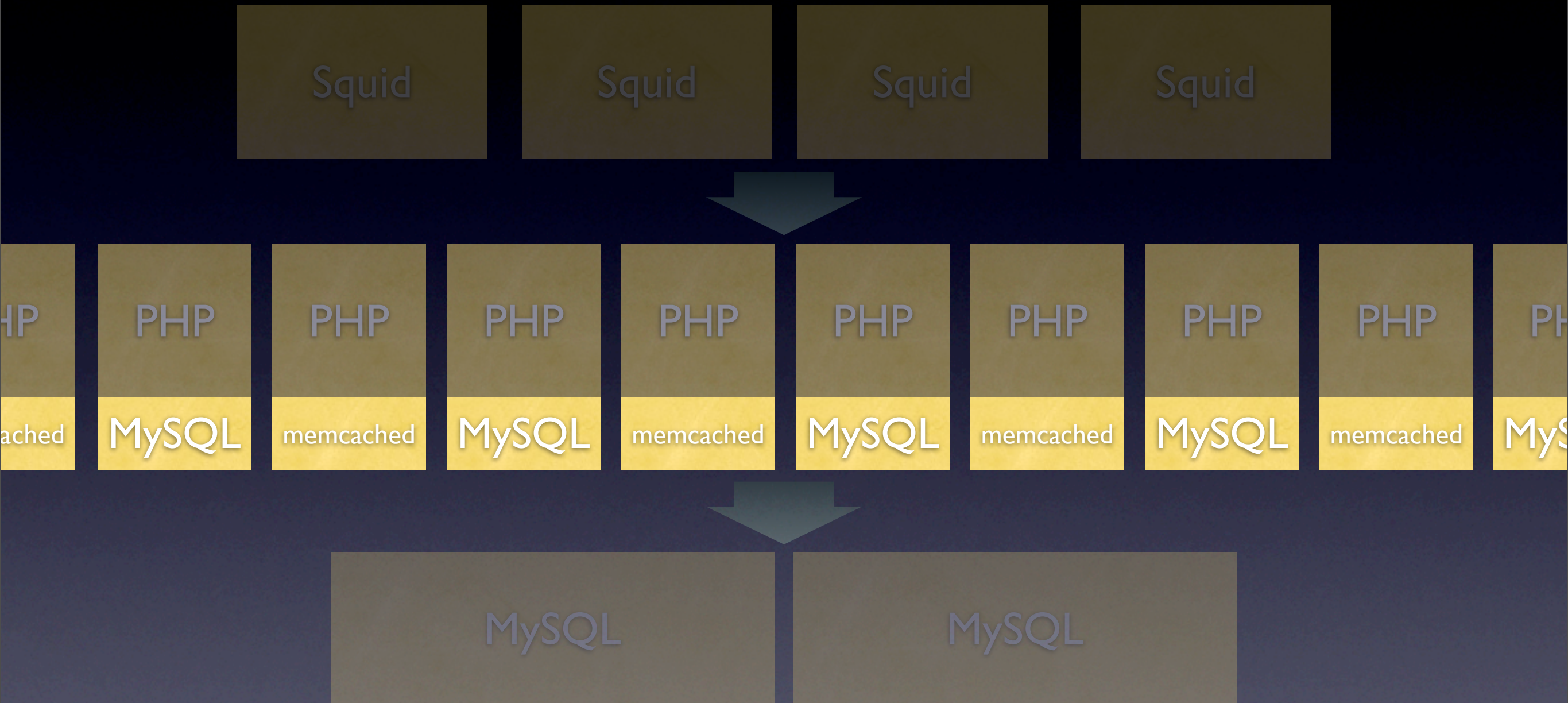
Faster than
disk-backed
database
when you
just need an
object
cache...


Now add cash!





Put underutilized memory and disk to work!



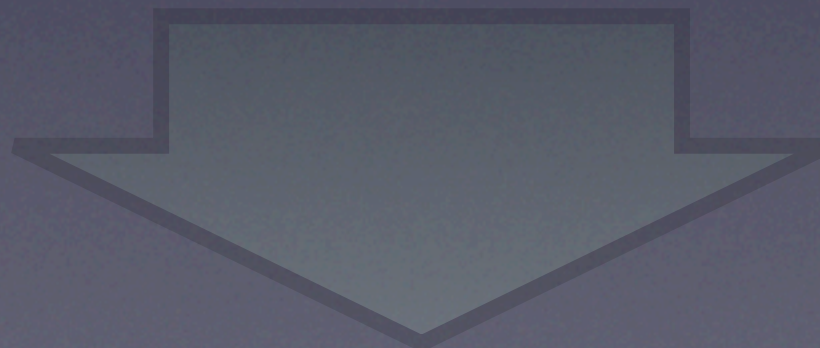


Those web servers come with
plenty of memory and disk space.
Use it!

memcached

MySQL

memc





HP

PHP

PHP

PH

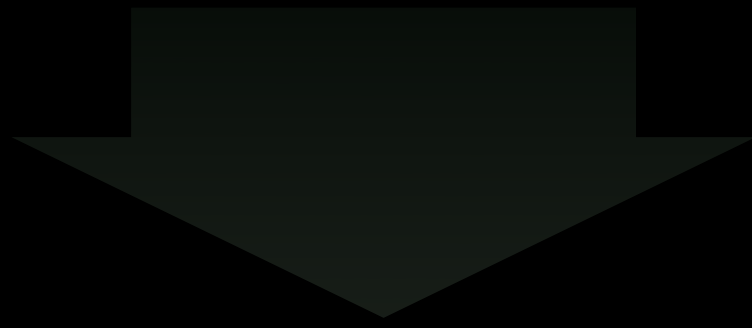
SQL

memcached

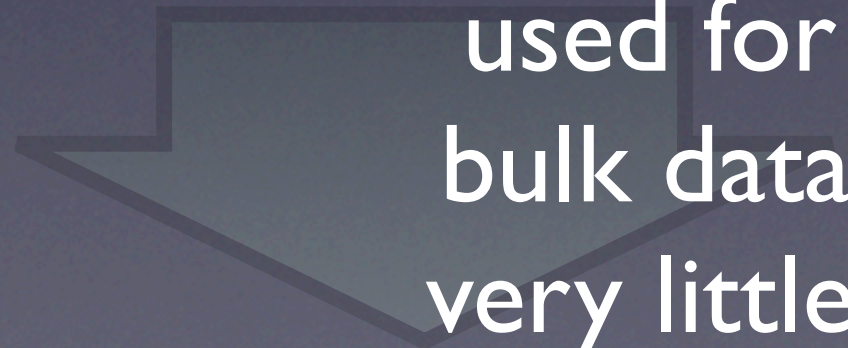
MySQL

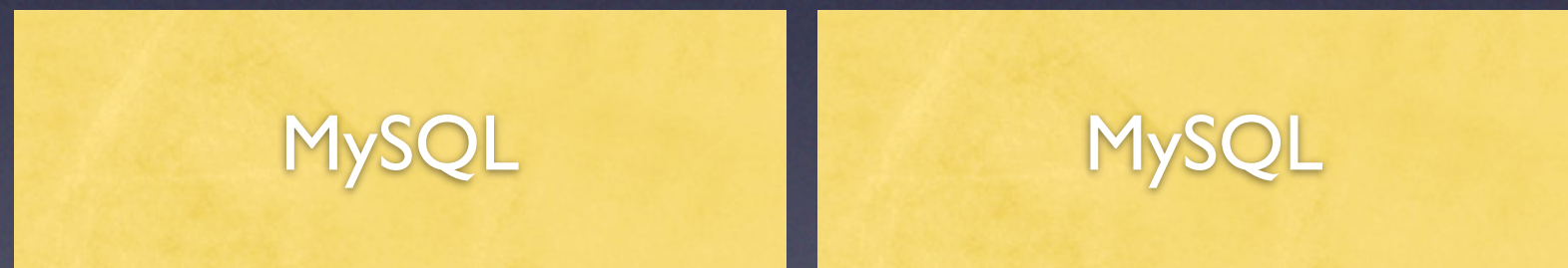
memc

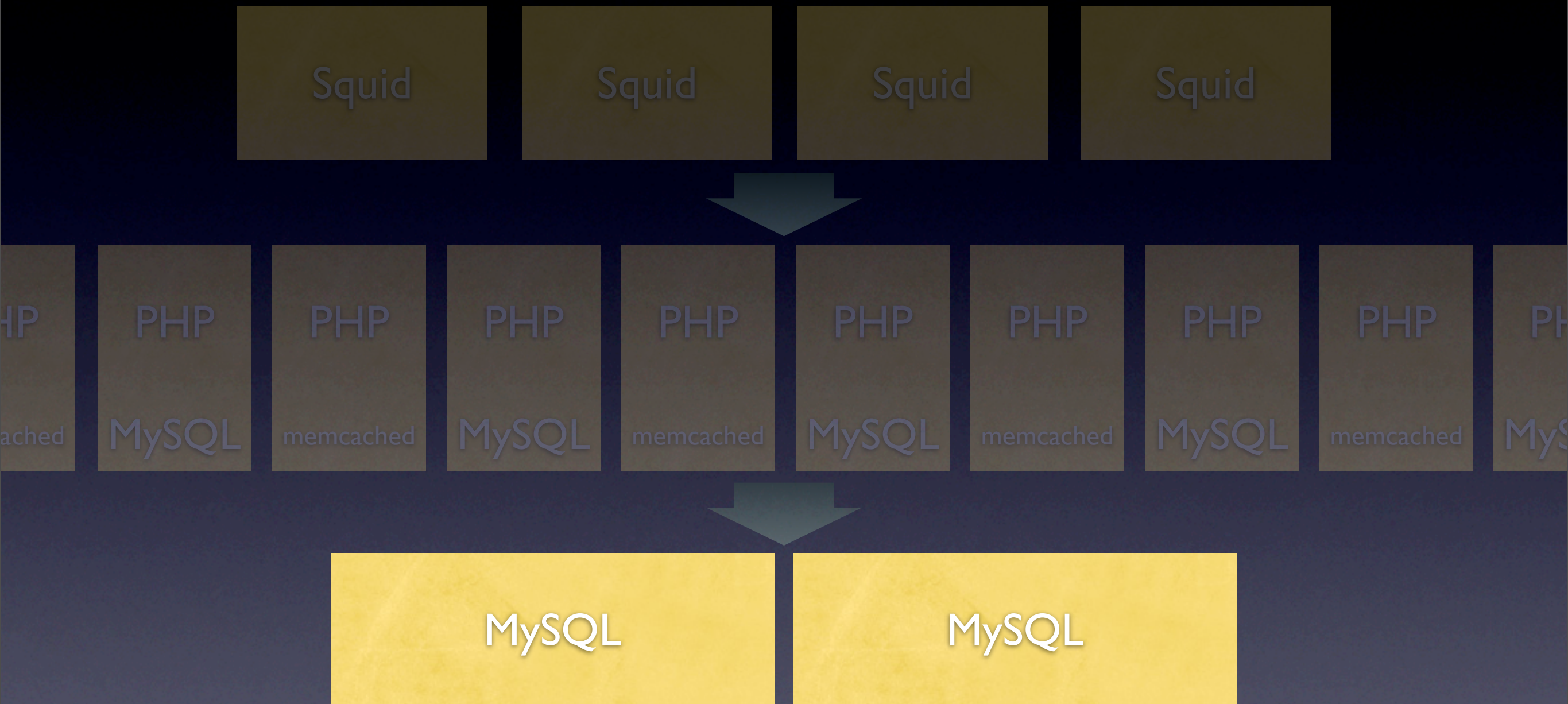
A bit of memory on
each server adds up
to a big memcached
store space...



Disk space can be
used for replicated
bulk data storage at
very little CPU cost.







Break it up...

Simple Sharding

Split along logical data partitions, such as subsites that don't interact closely.

MySQL group s1

English-language Wikipedia

MySQL group s2

Next 19 biggest wikis

MySQL group s3

Next 764 wikis

Functional Sharding

Split along functional boundaries...

MySQL big iron

Page metadata, links, users...
...read/write/update
...active index scans

PHP

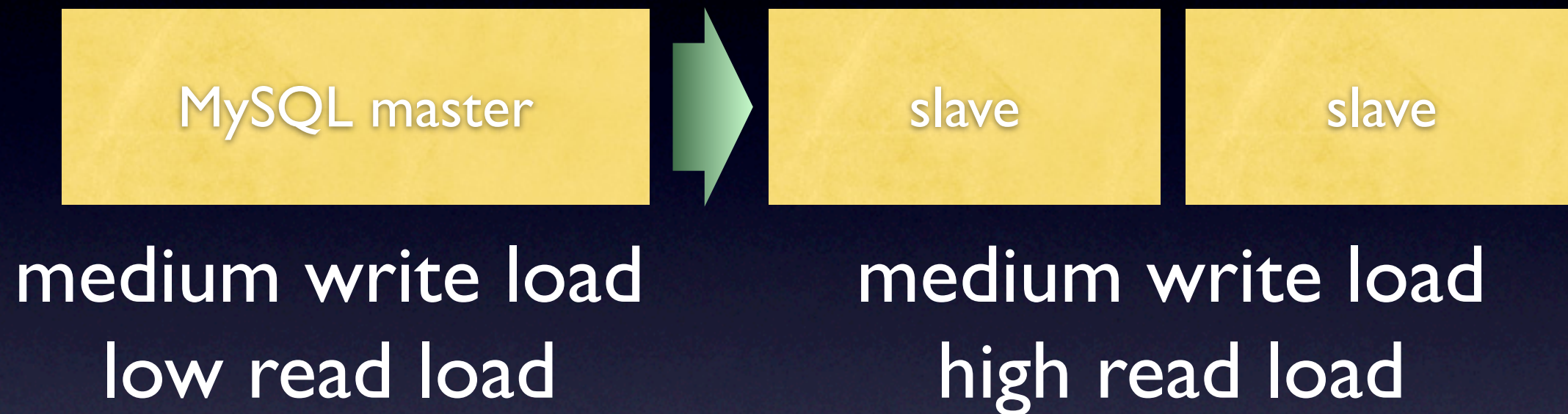
PHP

PHP

MySQL MySQL MySQL

Append-only bulk text
...nice simple blobs

Replicate for speed!



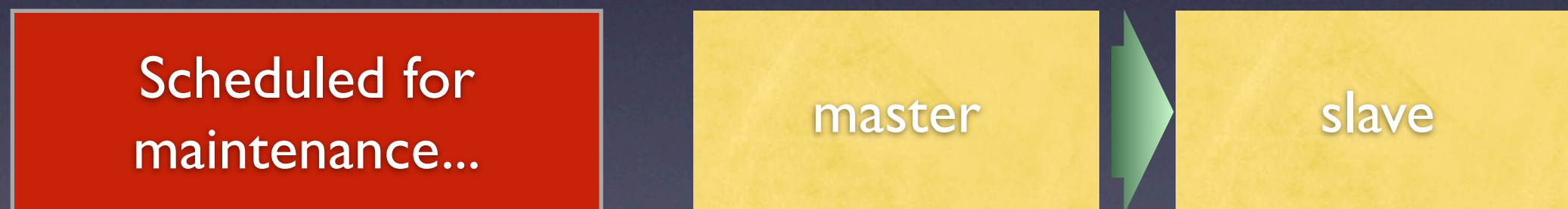
...but your application now has to think about replication lag.

& reliability...



Master dead?

Promote a slave!



...but failover isn't automated with MySQL.

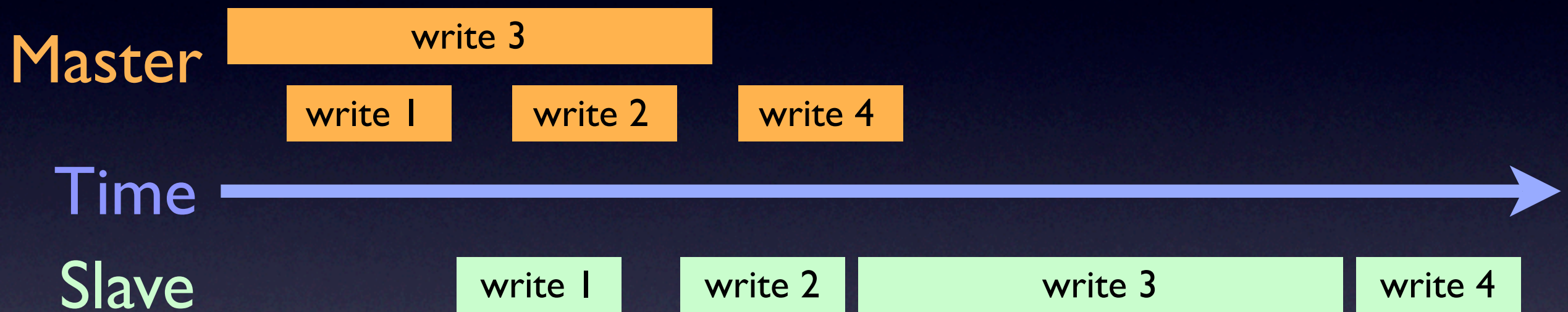
Replication lag?

Where'd my edit go???

The image illustrates the concept of replication lag using three overlapping screenshots of the Wikipedia article "LAMP (software bundle)".

- Top-left screenshot:** Shows the original article content. The text describes the acronym LAMP as referring to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers. The original expansion is as follows:
 - Linux, referring to the operating system;
 - Apache, the Web server;
 - MySQL, the database management system (or database server);
 - PHP or others, i.e. Perl, Python, the programming languages.The combination of these technologies is used primarily to define a web server infrastructure.
- Middle screenshot:** Shows the article in "Editing" mode. A warning message states: "Please do not save test edits. If you want to experiment, please use the sandbox." The edit area shows the text: "The [[acronym]] 'LAMP' refers to TOTALLY AWESOME SOFTWARE THAT DOES COOL STUFF" and "THIS IS MY EDIT IT IS BETTER THAN YOURRRRRSSSS".
- Bottom-right screenshot:** Shows the article after the edit, but the changes are not visible. The article content remains the same as in the top-left screenshot, illustrating replication lag.

Replication lag!



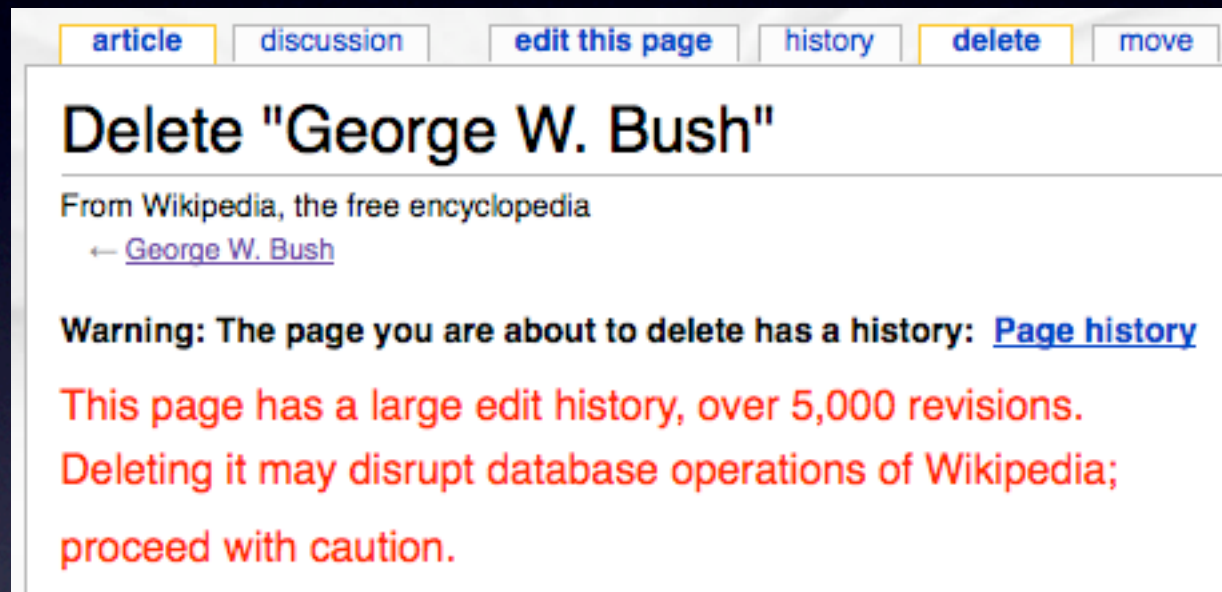
- Writes are replicated after they complete on the master...
- ...and parallel writes are serialized!

Choose hardware wisely

- **Conventional wisdom:**
Master should be beefy megaserver, slaves
can be wimpy
- **Secret truth:** slaves need to handle
writes **faster** than master to keep up!

Avoid slow writes!

- UPDATE or DELETE of thousands of rows



revision
🔑 rev_id: INTEGER(8)
🔑 rev_page: INTEGER(8) (FK)
🔑 rev_text_id: INTEGER(8) (FK)
💎 rev_comment: TINYBLOB
🔑 rev_user: INTEGER(5) (FK)
💎 rev_user_text: VARCHAR(255)
💎 rev_timestamp: CHAR(14)
💎 rev_minor_edit: TINYINT(1)
💎 rev_deleted: TINYINT(1)
💎 rev_len: INTEGER(8)
💎 rev_parent_id: INTEGER(8)

- Schema updates -- ALTER TABLE

Mass updates are a schema smell

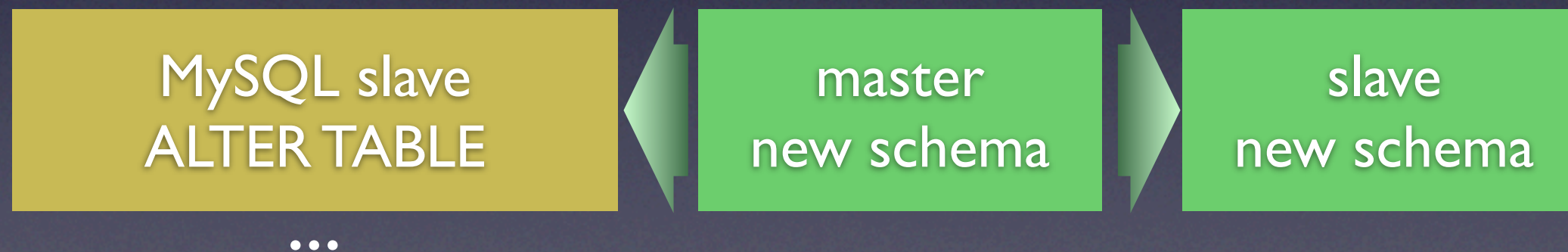
- **Hack:** Disable renaming for “George W. Bush” page so we don’t have to UPDATE title in 50,000 old-revision rows
- **Better:** run bulk UPDATEs in small batches to help the replication stream serialize
- **Best:** Refactor DB schema so title need only be updated once in article page table

Fun upgrade tricks



apply schema changes on slaves,
disabling them one at time...

...then swap masters...



...for low-downtime column and
index changes!

Application logic

- Check slaves for amount of lag -- application-specific fallback behavior
- Automatically engage maintenance mode if slaves are too far behind
- Store the last master position the session needs -- often we can catch up quickly!

Factory functions

Abstract your database connections!

```
$dbw = wfGetDB(DB_MASTER);  
$dbw->insert(...);
```

```
    $dbr = wfGetDB(DB_SLAVE);  
    $row = $dbr->selectRow(...);
```


wfGetDB(DB_MASTER)
LoadBalancer_Multi
connect to master...
DatabaseMySQL

wfGetDB(DB_SLAVE)
LoadBalancer_Multi
check slave lag...
find a slave caught up to session state...
DatabaseMySQL
report warnings if lag too high...

Session data!

```
/**
 * Notify the ChronologyProtector that the LBFactory
 * is done calling shutdownLB() for now.
 * May commit chronology data to persistent storage.
 */
function shutdown() {
    if ( session_id() != '' &&
        count( $this->shutdownPos ) ) {
        wfDebug( __METHOD__." : saving master pos for " .
            count( $this->shutdownPos ) .
            " master(s)\n" );
        $_SESSION[__CLASS__] =
            $this->shutdownPos;
    }
}
```


Questions?

<http://leuksman.com/pages/presentations>